



Connect an iPhone, iPad, or iPod touch to Arduino with the Redpark Serial Cable

Written By: Brian Jepson



PARTS:

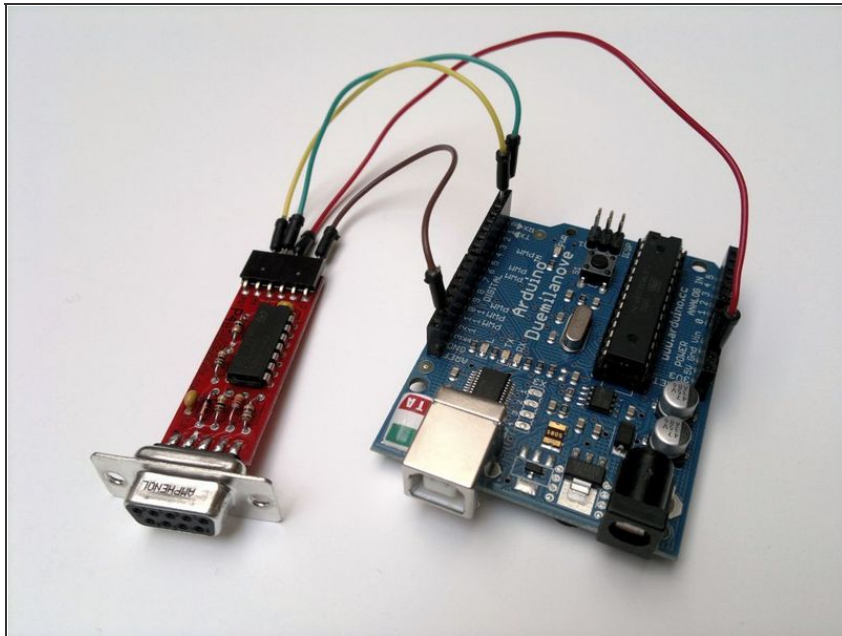
- [Breakout Pack for Arduino and iPhone \(1\)](#)
- [iPhone, iPad or iPod touch \(1\)](#)
- [Arduino Uno \(1\)](#)

SUMMARY

The Redpark Serial Cable, which is part of the [Redpark Breakout Pack for Arduino and iOS available from the Maker Shed](#), lets you connect the iPhone to Arduino without jail breaking. This guide shows you one of the simplest examples: connecting an iOS device to an Arduino and turning an LED on and off from an iOS app.

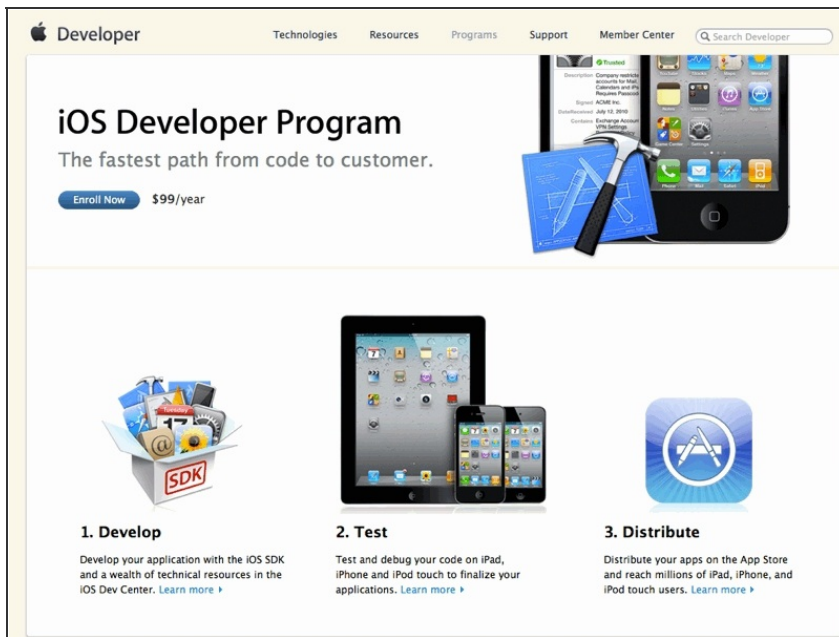
(This guide written with lots of help from [Alasdair Allan](#)), the author of a book on this very topic, [iOS Sensor Apps with Arduino](#).

Step 1 — Wire up the RS232 adapter to the Arduino



- Connect the RS232 adapter's power and ground to the Arduino's 5V and GND pins, respectively.
- Connect the RS232 adapter's TX to the Arduino's RX, and the RS232 adapter's RX to the Arduino's TX.


Step 2 — Make sure you're set up for iOS development



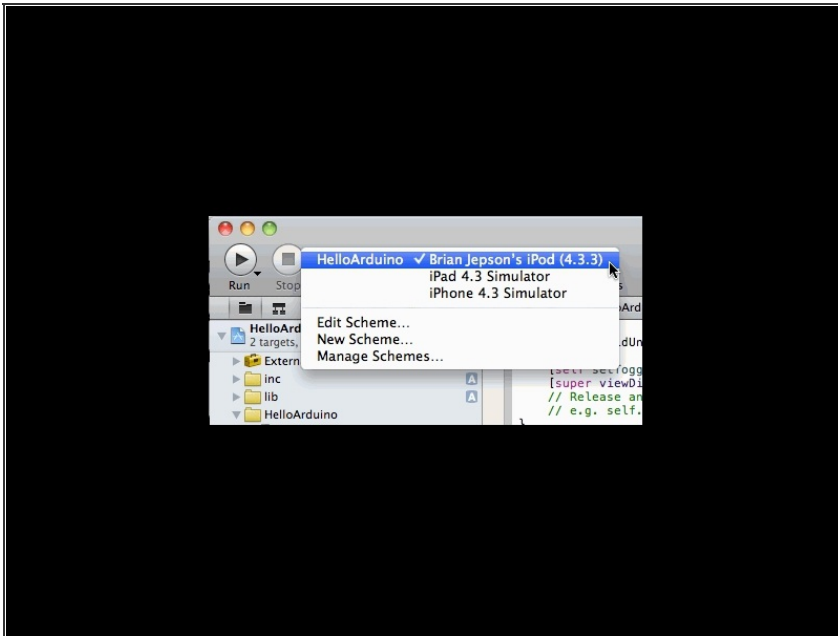
1. Develop
Develop your application with the iOS SDK and a wealth of technical resources in the iOS Dev Center. [Learn more](#)

2. Test
Test and debug your code on iPad, iPhone and iPod touch to finalize your applications. [Learn more](#)

3. Distribute
Distribute your apps on the App Store and reach millions of iPad, iPhone, and iPod touch users. [Learn more](#)

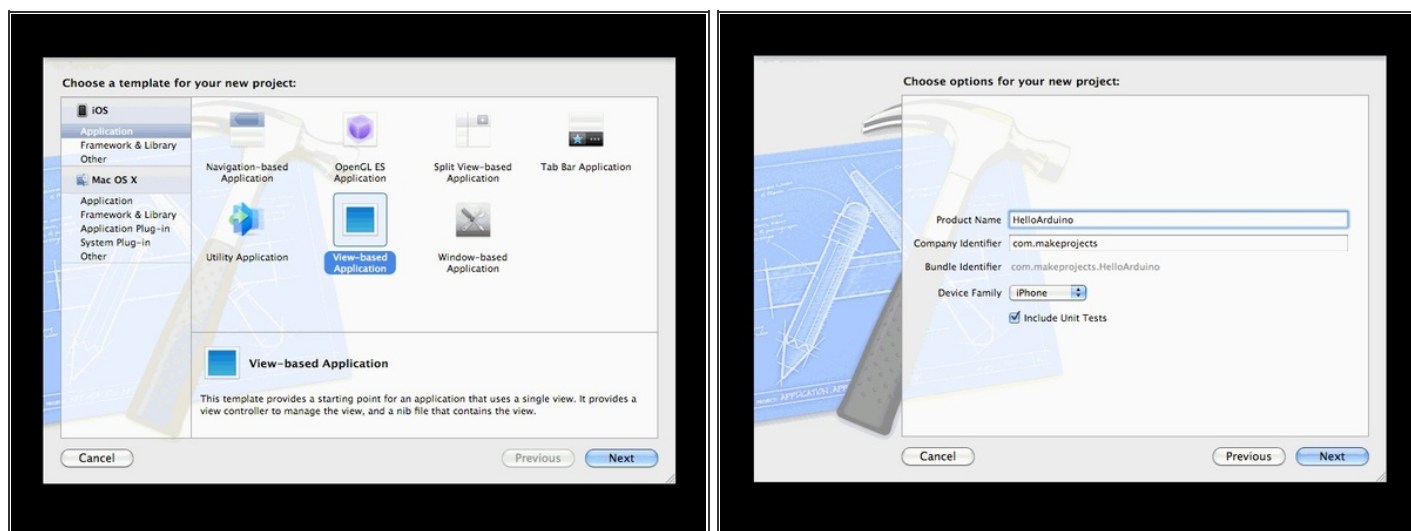
- To develop apps for the iPhone, iPad, or iPod touch, you'll need to register as an iOS developer. Although you can register for free, all that gets you is the development tools. You won't be able to deploy your apps to a real device. It's \$99 a year for an individual developer, \$299 for Enterprises (this option lets you deploy apps directly to your team), and free for educational institutions (which also lets you deploy apps to a team).
- For more information, see the [iOS Developer Program](#). 

Step 3 — Make sure you can run an app on your iOS device



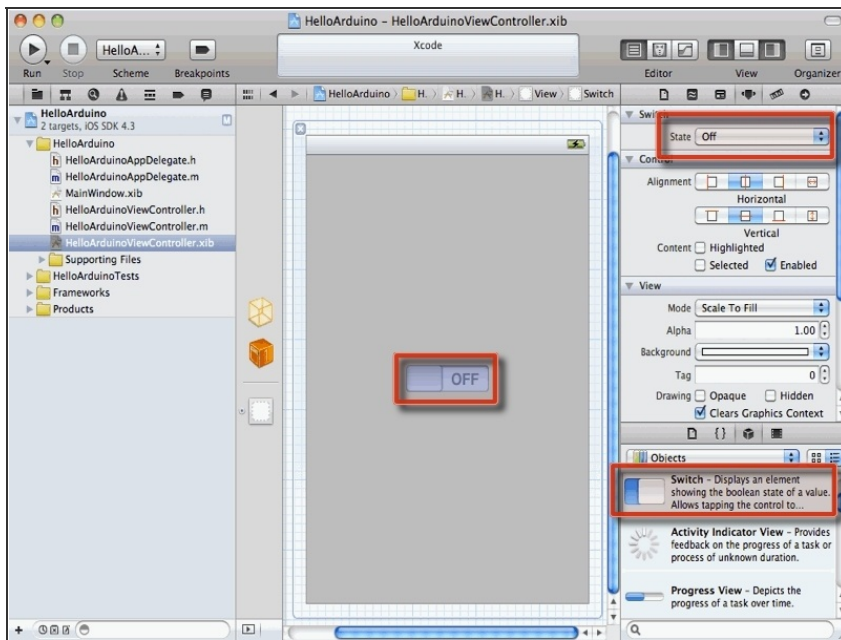
- If you've never programmed for the iPhone before, you can't count on this guide to help you. There are many learning resources, from Stanford's free [Developing Apps for iOS](#) courses to Matt Neuburg's [Programming iOS 5](#) or Alasdair Allan's [Learning iOS Programming](#), both of which are available in print or ebook form.
- Before you go any further, please make sure you can create and run simple apps on your iPhone, iPad, or iPod touch. It will also be helpful if you familiarize yourself thoroughly with the Xcode development environment.

Step 4 — Create a new view-based project



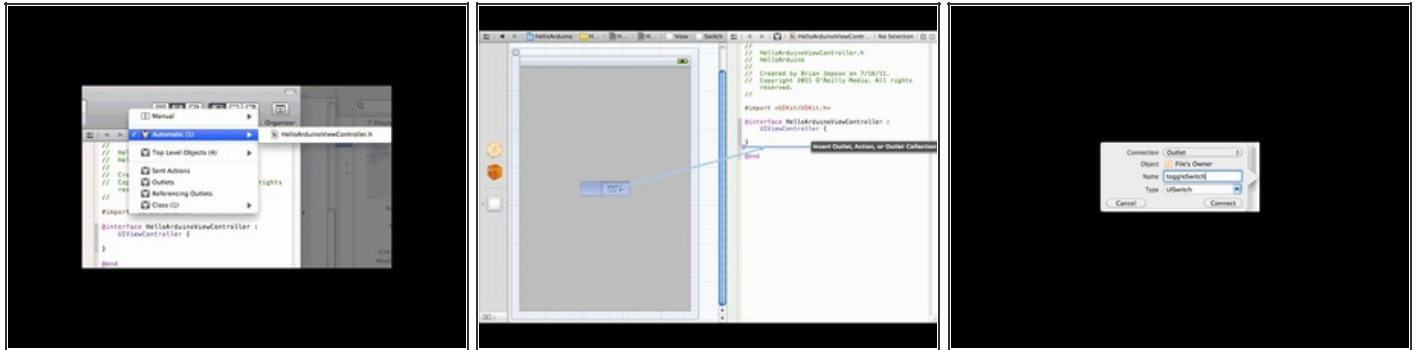
- In Xcode, create a new view-based project.
- Give your project the name HelloArduino, and choose the appropriate device family (iPhone, iPad, or if you're feeling ambitious, Universal, though this will create a bit more work for you).



Step 5 — Add a switch to your app



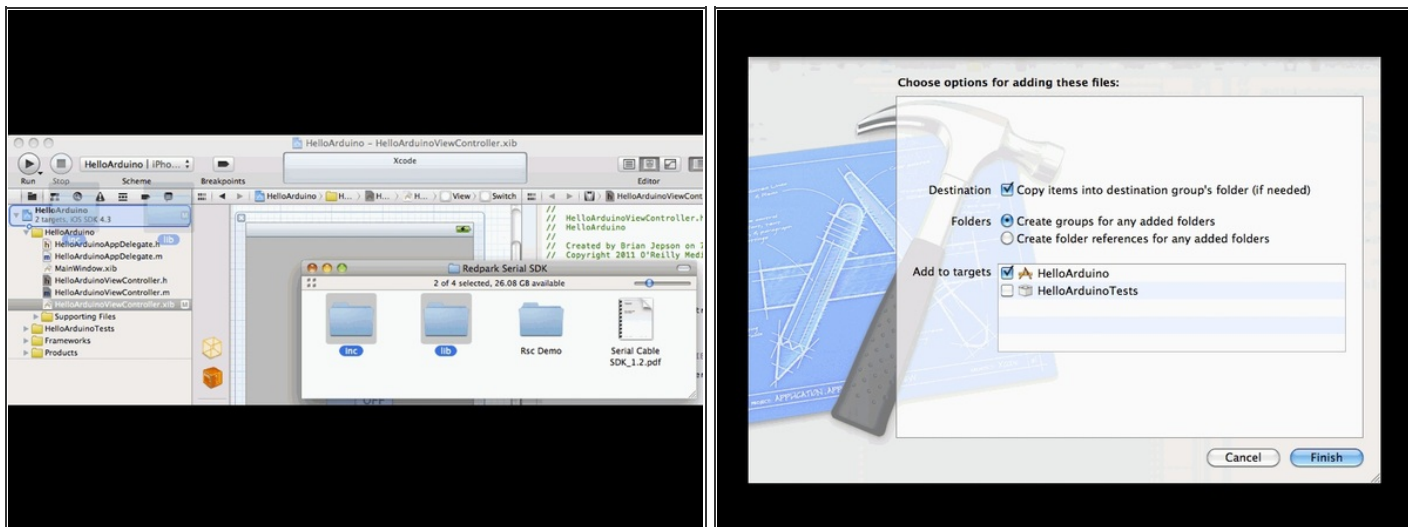
- On the left pane of the screen, expand your project, and open the folder HelloArduino.
- Next, locate the HelloArduinoViewController.xib and click on it. This will bring up your app's main view, which is blank at the moment.
- Bring up the Object Library (View→Utilities→Object Library), and drag a Switch to the center of your view.
- Next, show the Attributes Inspector (View→Utilities→Attributes Inspector) and set the switch's State to Off.



Step 6 — Wire up the switch to your code



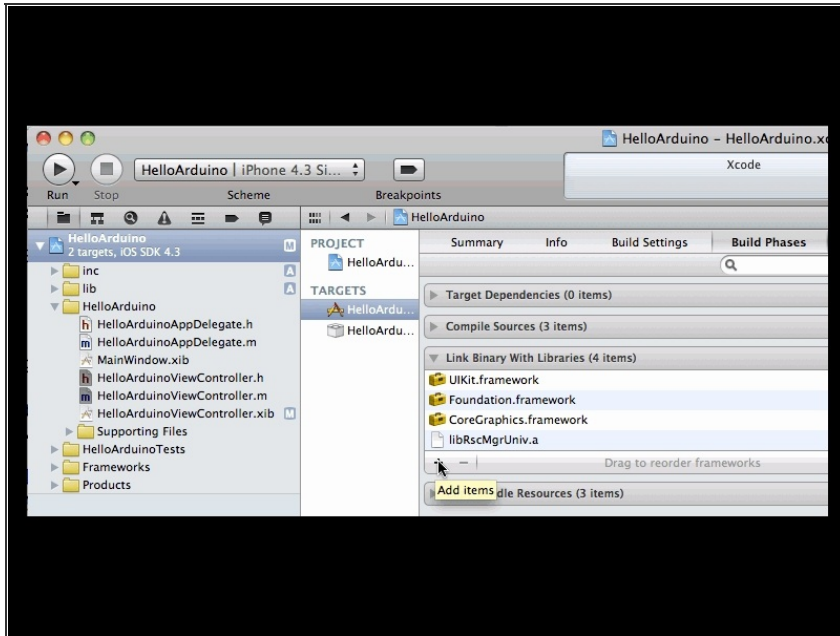
- Now you need to tell the app's code what to do with the switch.
- If you need room on the screen, you can hide the Utilities Pane (View→Utilities→Hide Utilities). 
- Next, open the Assistant Editor (View→Editor→Assistant), and locate the toolbar above the window that appears.
- Click the icon to the right of the left/right arrows and make sure the Assistant Editor is set to automatic. It should be showing the HelloArduinoViewController.h file.
- Control-click the switch, and drag from it to the code that appears in the Assistant Editor. Hold it just above the "@end" in the code and release. Add a new outlet and name it toggleSwitch as shown.
- Do this once again, but instead, add a new action, and name it toggleLED. Save the file (File→Save). 

Step 7 — Import the Redpark Serial Cable library



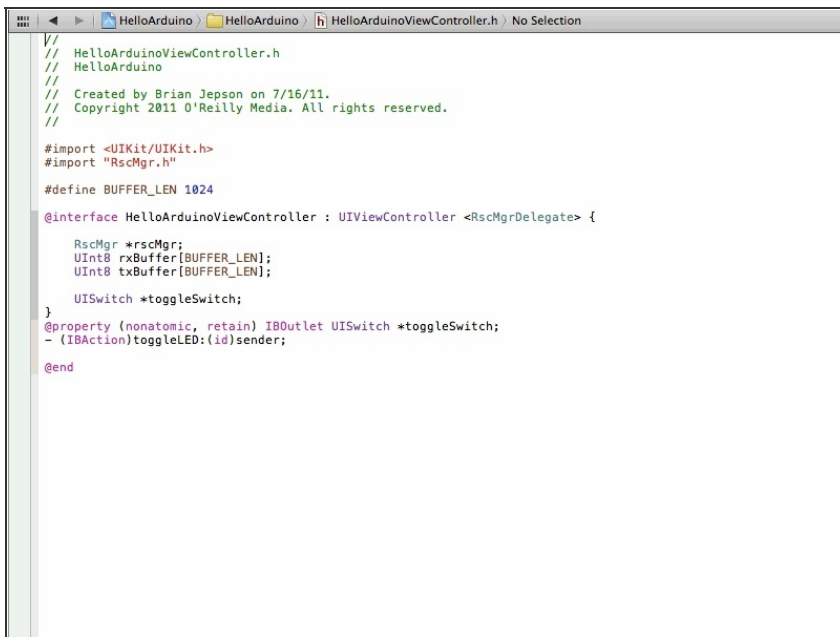
- The Redpark Serial Cable installation will create a folder in your home directory named Redpark Serial SDK. 
- You can download the SDK from [the Redpark web site](#). 
- Open this in the Finder, and select the inc and lib folders, then drag them to your project as shown.
- A dialog will appear; make sure you check the option "Copy items into destinations group's folder (if needed)" and click Finish.

Step 8 — Import the iOS Accessory Framework



- Next, you need to import Apple's Accessory Framework. To do this, click the Project in the left pane, then choose Build Phases, and open the "Link Binary With" section. Click +, locate the ExternalAccessory.framework, and add it.

Step 9 — Add some declarations to the view controller



- Select the HelloArduinoViewController.h file to open it for editing.
- Visit my modified version of the file at [GitHub](#).
- Edit your copy of the file:
 - Add the two lines between `#import <UIKit/UIKit.h>` and `@interface`
 - Modify the `@interface` line to match mine.
 - Add the three lines just above `UISwitch *toggleSwitch;.`
- Save the file (File→Save).

Step 10 — Add some functions to the app delegate

```

HelloArduino HelloArduino HelloArduinoViewController.m No Selection
//
// HelloArduinoViewController.m
// HelloArduino
//
// Created by Brian Jepsen on 7/16/11.
// Copyright 2011 O'Reilly Media. All rights reserved.
//

#import "HelloArduinoViewController.h"

@implementation HelloArduinoViewController
@synthesize toggleSwitch;

- (void)dealloc
{
    [toggleSwitch release];
    [super dealloc];
}

- (void)didReceiveMemoryWarning
{
    // Releases the view if it doesn't have a superview.
    [super didReceiveMemoryWarning];


    // Release any cached data, images, etc that aren't in use.
}

#pragma mark - View lifecycle

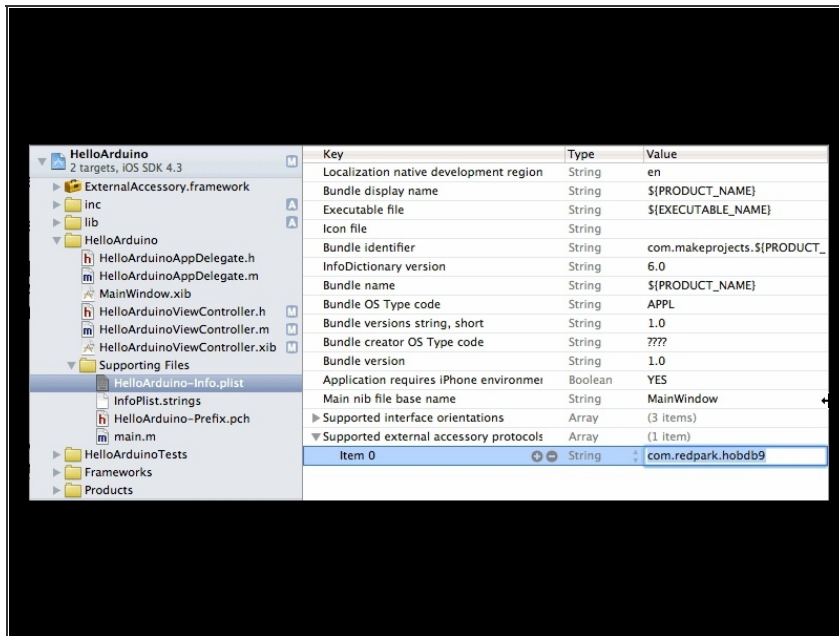
// Implement viewDidLoad to do additional setup after loading the view, typically from a nib.
- (void)viewDidLoad
{
    [super viewDidLoad];
    rscMgr = [[RscMgr alloc] init];
    [rscMgr setDelegate:self];
}


- (void)viewDidUnload
{
    [self setToggleSwitch:nil];
    [super viewDidUnload];
    // Release any retained subviews of the main view.

```

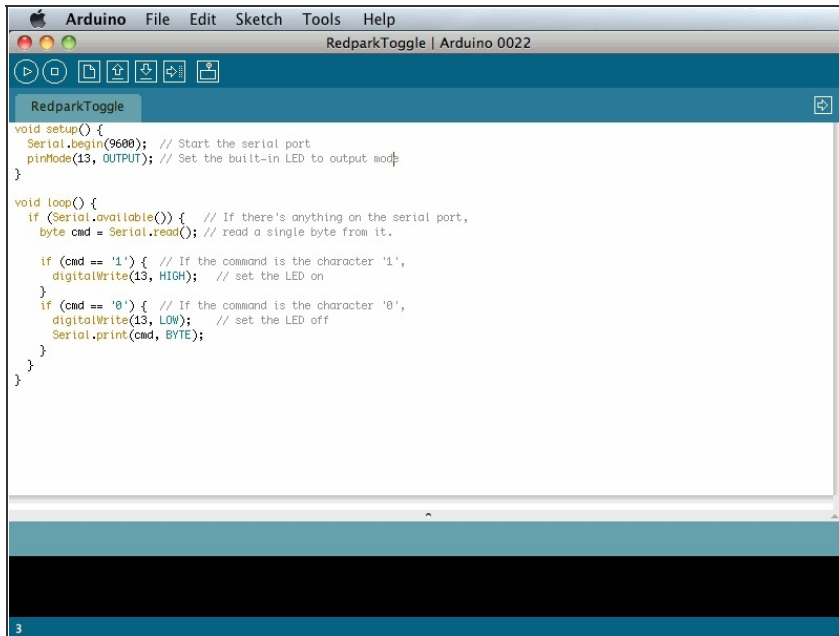
- Select the HelloArduinoViewController.m file to open it for editing.
- Visit my modified version of the file at [GitHub](#) then edit your copy of the file:
 - Locate the viewDidLoad method and replace it with my version of it.
 - Be sure to Remove the comment delimiters  (* and */) before and after it.
 - Implement the methods required by the RscMgrDelegate protocol (add everything from my file from the #pragma mark - RscMgrDelegate methods to the end of the file.
 - Locate the toggleLED method and replace it with my version of it.
- Save the file (File→Save).

Step 11 — Declare support for the serial cable




- This step is optional, but it will avoid the problem of you  seeing an error message ("This accessory requires an application...") each time you plug in the cable.
- In Xcode, expand the Supporting Files group and click on Hello-Arduino.plist to open it.
- Right-click the bottom row, and choose Add Row. Click the up/down pointing arrows to the right of the new row's key (it will probably default to "Application Category") and choose "Supported external accessory protocols".
- Click the triangle to the left of the key name you just selected to open up the list. In the value field for Item 0, type `com.redpark.hobdb9`.
- Save the file (File→Save).

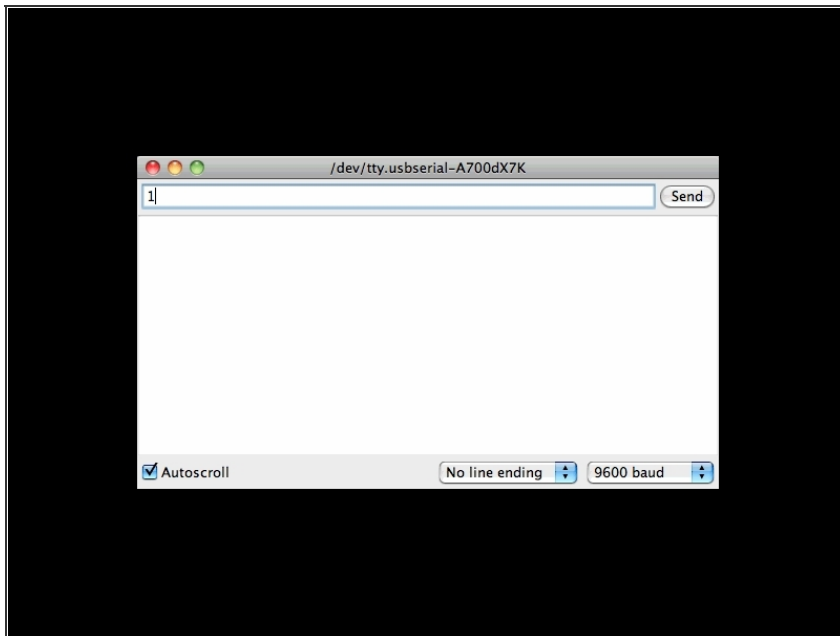
Step 12 — Deploy and test the Arduino sketch

A screenshot of the Arduino IDE interface. The window title is "RedparkToggle | Arduino 0022". The code editor shows the following C++ code:

```
void setup() {  
  Serial.begin(9600); // Start the serial port  
  pinMode(13, OUTPUT); // Set the built-in LED to output mode  
}  
  
void loop() {  
  if (Serial.available()) { // If there's anything on the serial port,  
    byte cmd = Serial.read(); // read a single byte from it.  
  
    if (cmd == '1') { // If the command is the character '1',  
      digitalWrite(13, HIGH); // set the LED on  
    }  
    if (cmd == '0') { // If the command is the character '0',  
      digitalWrite(13, LOW); // set the LED off  
      Serial.print(cmd, BYTE);  
    }  
  }  
}
```

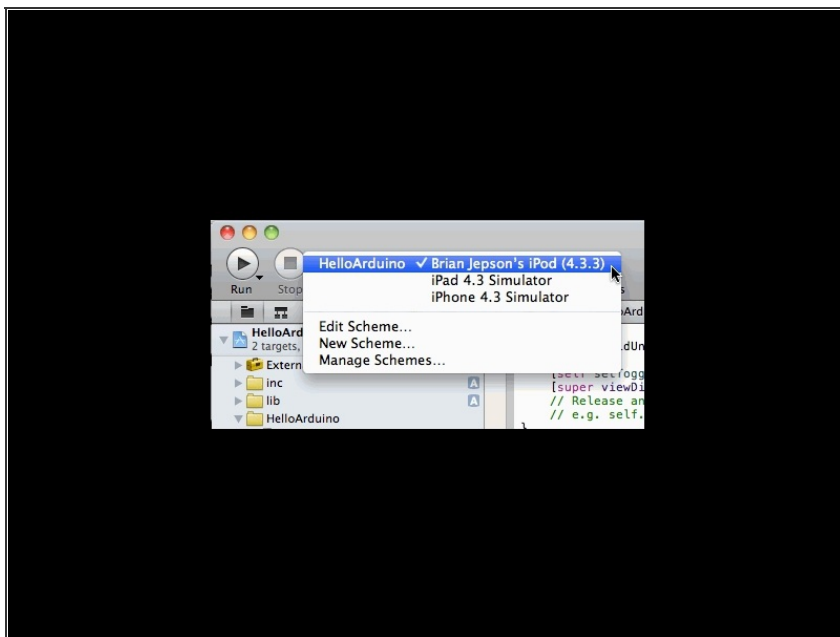
- Grab a copy of the sketch from [GitHub](#).
- Connect your Arduino to your computer.
- Run the Arduino IDE and upload the RedparkToggle sketch to your Arduino.
- Temporarily disconnect the jumper wire going to the Arduino's RX pin (the green cable shown in the first step), and upload your sketch.
- You must disconnect it before uploading a sketch  because the RS232-TTL adapter will otherwise interfere with the upload process. Leave the jumper wire disconnected until you complete the next step.


Step 13 — Test the Arduino



- Open the Arduino's serial monitor (Tools→Serial Monitor), and make sure it's set to 9600 baud.
- Wait a few seconds for the sketch to restart, then type 1 and click Send. The LED should come on. Try that again with 0 instead of 1, and the LED should go off.
- Reconnect the jumper wire when you're done.

Step 14 — Deploy the app



- Connect your iPhone, iPad, or iPod touch to your Mac with a dock cable.
- Make sure Xcode is configured to run it on the device you just connected (see the figure).
- Click the Run button in Xcode.
- If you get any errors, review the preceding steps to make  sure you followed them exactly. If you still have problems, you can try downloading the project from [GitHub](#).

Step 15 — Run the app



- The app is running on your iOS device, but you need to swap cables next, so click Stop in Xcode and disconnect the dock cable.
- Plug the Redpark Serial Cable into your device and into the RS232 serial adapter.
- Locate the HelloArduino app on the home screen, and run it.

If everything is working correctly, you'll be able to turn the Arduino's built-in LED on and off by tapping the switch button!

This document was last generated on 2012-11-03 03:13:04 AM.